Jméno: **UČO:**

0007 *list* 1 *učo* *body*

Oblast strojově snímaných informací. Své učo a číslo listu vyplňte zleva dle vzoru číslic. Jinak do této oblasti nezasahujte. 0 1 2 3 4 5 6 7 8 9

1. [2 body] Nechť $\Sigma = \{l, u, f, o, g\}$. Navrhněte algoritmus, který pro zadaný NFA $\mathcal{A}$ pracující s abecedou Σ zkontroluje zamykání zámku v jednotlivých slovech akceptovaných automatem $\mathcal{A}$. Kontrolu budeme provádět na slovech, v nichž l představuje akci „zamknout zámek“ (*lock*) a u představuje akci „odemknout zámek“ (*unlock*), písmena f, g a o představují akce, které na zámek nemají vliv.

Chceme kontrolovat, že

- po každém zamknutí dojde k odemknutí zámku,
- nepokoušíme se odemknout již odemčený zámek,
- nepokoušíme se zamknout již zamčený zámek,
- na konci je zámek odemčený.

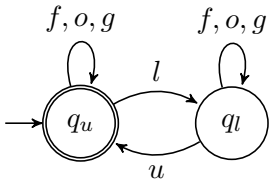
Předpokládejme, že začínáme s odemčeným zámkem. Příklady slov, v nichž je manipulace se zámky v pořádku, jsou například ε, $lfuog$, gfg, $glulguf$, lu, $glou$, naopak podmínky nesplňují například slova uff, $lluu$, $lufgl$, $glulgufu$, ufo, lol, gl. Algoritmus musí vrátit *true*, pokud všechna slova akceptovaná automatem $\mathcal{A}$ pracují se zámky správně podle uvedených pravidel, v opačném případě musí vrátit *false*.

Přesně popište princip fungování Vašeho algoritmu (nemusíte nutně použít pseudokód) a dokažte, že je tento algoritmus konvergentní (vždy skončí). Můžete využívat libovolné algoritmy z přednášky, pokud na to v textu patřičně upozorníte (nestačí však odkazovat se na dokázaná tvrzení, v jejichž důkazu není algoritmus).

Požadovaná vlastnost správného zamykání zámku se na první pohled zdá být docela jednoduše popsatelná. Zásadní myšlenka je, že pokud bychom tuto vlastnost dokázali vyjádřit pomocí regulárního jazyka, mohli bychom následně kontrolu zvládnout za pomoci známých operací nad regulárními jazyky, respektive konečnými automaty. Cílem bude tedy vyjádřit jazyk všech slov se správným zamykáním a následně otestovat, zda je jazyk vstupního automatu podmnožinou tohoto jazyka.

Náš hledaný jazyk by měl být *nejobecnějším* jazykem majícím požadovanou vlastnost v uvedené abecedě. To znamená, že bude obsahovat právě a jenom všechna možná slova se správným zamykáním zámku. Označme ho $L_{lock} \subseteq \Sigma^*$. Zároveň by tento jazyk měl být regulární, aby se nám s ním dále dobře pracovalo.

L_{lock} bude velice užitečný proto, že náš algoritmus by měl vracet *true* právě tehdy, když $L(\mathcal{A}) \subseteq L_{lock}$, neboli $L(\mathcal{A}) \setminus L_{lock} = \emptyset$. Jak brzy uvidíme, L_{lock} se nám bude hodit popsaný DFA $\mathcal{M}$ daným následujícím přechodovým grafem:



Platí $L(\mathcal{M}) = L_{lock}$, protože automat $\mathcal{M}$ kontroluje právě podmínky zamykání, tedy akceptuje jakékoli slovo $w \in \Sigma^*$ takové, které je splňuje, žádné jiné podmínky na w neklade. S automatem $\mathcal{M}$ po ruce definujeme náš algoritmus následovně.

Jméno: **UČO:**

0007 *list* 2 *učo* *body*

Oblast strojově snímaných informací. Své učo a číslo listu vyplňte zleva dle vzoru číslic. Jinak do této oblasti nezasahujte. 0123456789

1. Determinizujeme $\mathcal{A}$ podle algoritmu z přednášky 4, slide 8, $\mathcal{A}$ po determinizaci označme $\mathcal{A}_{det}$.
2. Zkonstruujeme synchronní paralelní kompozici $\mathcal{A}_{diff} = \mathcal{A}_{det} \setminus \mathcal{M}$ podle algoritmu z přednášky 2, slide 11.
   - Postupujeme jako pro průnik, jen výsledná množina akceptujících stavů je $F_3 = F_1 \times (Q_2 \setminus F_2)$. Tedy $\mathcal{A}_{diff}$ akceptuje právě ve stavech, jejichž první složka odpovídá akceptujícímu stavu $\mathcal{A}_{det}$ a druhá složka neakceptujícímu stavu $\mathcal{M}$. Tedy zjevně $\mathcal{A}_{diff}$ akceptuje právě slova, která $\mathcal{A}_{det}$ akceptuje a zároveň $\mathcal{M}$ zamítá.
3. V $\mathcal{A}_{diff}$ eliminujeme nedosažitelné stavy podle algoritmu z přednášky 4, slide 4. Dostáváme $\mathcal{A}'_{diff}$.
4. Nechť F je množina koncových stavů $\mathcal{A}'_{diff}$ (t.j. F je množina dosažitelných koncových stavů $\mathcal{A}_{diff}$).
5. Vrátíme *true*, pokud $F = \emptyset$, jinak vrátíme *false*.

Korektnost algoritmu vyplývá z toho, co bylo řečeno již na začátku. Víme, že $L(\mathcal{M}) = L_{lock}$, a jelikož determinizace je ekvivalentní úprava, rovněž platí $L(\mathcal{A}_{det}) = L(\mathcal{A})$. Synchronní paralelní kompozice $\mathcal{A}_{diff}$ potom z definice akceptuje jazyk $L(\mathcal{A}_{diff}) = L(\mathcal{A}_{det}) \setminus L(\mathcal{M}) = L(\mathcal{A}) \setminus L_{lock}$.

Pokud ukážeme, že kompozice $\mathcal{A}_{diff}$ nemá žádný dosažitelný koncový stav, tedy $F = \emptyset$, z toho hned plyne, že $L(\mathcal{A}) \setminus L_{lock} = \emptyset$. To znamená, že $L(\mathcal{A})$ nemůže obsahovat žádná slova porušující pravidla zamykání, protože taková slova by nebyla odstraněna odečtením L_{lock} a náš algoritmus tedy vrátí korektně *true*.

Pokud $F \neq \emptyset$, zjevně $L(\mathcal{A}) \setminus L_{lock} \neq \emptyset$, tedy v $L(\mathcal{A})$ nutně existují slova, která porušují vlastnost zamykání zámku, protože jinak by byla i v L_{lock}. Algoritmus tedy opět správně vrací *false*. Tím jsme dokázali korektnost.

Náš algoritmus je konvergentní, protože algoritmy 1., 2. a 3. jsou konvergentní a naše práce s těmito algoritmy neobsahuje žádné cykly.